NASA-TM-112326

NDB
IN 60-TM

AIAA paper 81-2124

CASE STUDY: USE OF THE MICROCOMPUTER AS A SCIENTIST'S PERSONAL WORKBENCH

V. R. Watson*
Ames Research Center, NASA, Moffett Field, California

and

W. A. VanCamp† and G. L. Villere†
Informatics Inc., Palo Alto, California

Abstract

A preliminary field test in which a scientist was provided with a Terak 8510a microcomputer for use as a personal office computer and as an interface to larger computers demonstrated a substantial improvement in user efficiency. As a result, such systems were judged to be cost effective, and 40 of these microcomputers were recently obtained for utilization by a variety of scientists and programmers involved in a broad range of applications. In addition, a VAX 11/780 computer supporting a comparable user community via VT100 terminals has recently been installed. Results of a study on the effectiveness of these two approaches for improving the efficiency of these user groups are presented.

* Scientist. Associate Fellow AIAA.

† Member of the Senior Staff.

§ 1.0 Introduction

The purpose of this paper is to report the results of an effort to improve the efficiency of scientists and programmers by providing them with either microcomputers to use as personal workbenches or terminals connected to a time-sharing computer.

In 1978, as a test of the workbench concept, a Terak 8510a microcomputer was installed as a workbench for a scientist who was developing atmospheric simulation codes for execution on a CDC 7600. This scientist had previously been using punched cards for input and computer printout for output. The workbench substantially improved the efficiency with which this scientist interacted with the CDC 7600 by providing a more efficient means for creating the codes and for evaluating the output. A screen oriented editor on the microcomputer was used to generate the code and a graphical display program (which permitted perusing graphical output off line from the CDC 7600) was used to help evaluate the output. The workbench also provided help with documenting the program and reporting the results. In addition, new techniques could be tested by executing scaled down models on the workbench in a stand-alone mode.

In 1980, approximately 40 additional Terak microcomputers were purchased for use as workbenches by programmers and scientists assigned to a wide variety of tasks. In 1981, a comparable user community was provided with VT100 terminals connected to a VAX 11/780 computer. This paper reports on the effectiveness of both environments in improving user efficiency and productivity.

§ 2.0 Description of Test Environments

The main computing resources at Ames Research Center, NASA, which were used during this survey consisted of a CDC 7600, several VAX 11/780 computers, and several dozen PDP11 minicomputers. The PDP11 and VAX computer systems were interconnected in a distributed network, using Digital's DECnet software and relatively high speed communications hardware. Most communication links were implemented with either 45 kbps or 1 Mbps synchronous interfaces. The DECnet network provided full route-through capability, so that each networked computer could access any other.

The CDC 7600 computer played a special role in the test of the scientist's workbench, since it provided the primary computing resource for production runs of the scientist's large computer models. A dedicated PDP11 was used as a front end "station" for the CDC 7600. This PDP11 station provided a CDC 7600 job submission facility for DECnet users and also provided file staging between the DECnet systems and CDC 7600.

§ 2.1 Description of Microcomputer/Workbench Environment The workbench environment was based upon the Terak 8510a microcomputer with two floppy disk drives. The standard configuration consisted of an LSI-11 processor with 56K bytes of main memory, 500K bytes of disk storage and a 320 by 240 pixel bit-mapped graphics display. Each Terak was connected to either a PDP11 or VAX 11/780 node in the DECnet network via a 9600 baud asynchronous terminal communication line. Typically, each PDP11 node provided network access to the CDC 7600, common disk storage, and printer/plotter facilities for four to eight Teraks.

Software on the Terak consisted of the UCSD Pascal operating system, together with locally written VT52 and Tektronix 4010 terminal emulator programs. Also provided, was locally written file transfer software, which could conveniently transfer text or binary files at throughput speeds of about one fourth of the 9600 baud communication rate.

§ 2.2 Description of the Time-Sharing Environment The time-sharing environment consisted of VT100 terminals connected to VAX 11/780 minicomputers. The VAX computers were located at a central site, with 9600 baud terminal communications provided by short-haul modems between buildings, and 300 and 1200 baud access through dial-up facilities. Each system supported an average of 40 VT100 terminals. Several of the VT100s used in the survey had been modified to include 640 by 480 pixel graphics capability with Tektronix PLOT-10 compatibility.

This relatively standard timesharing environment utilized Digital's VMS operating system. The standard VAX/VMS editor, EDT, provided line or screen oriented text editting, with a convenient set of editting commands, but only rudimentary macro command facilities. The VAX/VMS process scheduler was modified at Ames to guarantee a fixed amount of CPU time to each research group that owned a "share" of a VAX 11/780. This "share scheduler" localized system saturation within individual research groups, allowing multiple groups to use the same system with relatively little intergroup conflict.

§ 3.0 Description of User Tasks and Work Habits

All of the users interviewed for this survey were engaged in applications that could be classified as either modeling or simulation. Typical areas of investigation included atmospheric modeling, calculations of the electronic properties of small molecules, and computational fluid dynamics. The programs in question ranged in size from 3K to 100K source lines (usually FORTRAN). Most of the programs in the survey produced large (in excess of 200K bytes) output files consisting of numerical data displayed in conventional tabular form. Users reported that when graphical output was employed, it was generally restricted to simple x-y plots or contour plots, and these were generated only after first examining the numerical output.

Whenever feasible, editing of source files was attempted without segmentation. However, for the larger programs this process often proved to be impractical because of poor editor/system performance, low transfer rates, or low disk capacity. In these situations, users were forced to segment their source files into smaller, more easily manipulated, components. Such segmentation operations were carried out at either the central computer facility or at the local network node. In the latter case, the user was still forced to absorb the overhead associated with transferring the entire source file between the central facility and the local node.

In order to peruse their nongraphical output files, users were forced to choose either rapid, unidirectional transfer of the file contents to the terminal screen, or use of the text editor to facilitate scrolling in both directions. In light of the previously mentioned difficulties involved with editing large files and because output files are not as amenable to segmentation as source files, most users opted to employ the uni-directional transfer technique. With regard to output line length, only users in the VT100 environment had any method available for dealing with standard 132-column output; the intelligent-terminal users had to accept either wraparound or truncation of the line beyond eighty characters.

Relatively little use was made of the graphics capabilities of either type of terminal, although most of the users surveyed acknowledged the importance (in principle) of graphical output. Most of the users who did generate graphical output for viewing at their terminals used either a dumb terminal or a Tektronix terminal emulator. Only two of the users surveyed had used the local processing capabilities of the intelligent terminals to process and display graphics files returned from the central computer facility.

§ 4.0 Results

The users of workbenches and the time-shared computers reported that they were much more efficient in these environments than in the previous environment which consisted of keypunches for preparing source code and line printers for obtaining computer output.

The study of workbench users indicated that the increase in efficiency gained by using the workbenches was highly user and problem dependent. Some users were able to significantly improve their efficiency, but other users did not use many of the features of the workbench and would have been just as efficient or even more efficient in the time-sharing environment.

The increase in efficiency of the users of the time-shared computers was more uniform than for the workbench users. Most of these users were able to learn how to use the system quickly, and, therefore, they utilized most of the features of this system. As a consequence, in the beginning of the study period, the efficiency of a typical user of this system was probably higher than the efficiency of most of the users of the workbenches. However, during the end of the study period, the time-shared computer system became "saturated" and users reported a significant decrease in their efficiency because of lengthy delays in response.

§ 4.1 Effectiveness for Specific Tasks

WORKBENCH APPROACH

— very effective for creating and modifying source code if the codes were segmented into modest size sections. There was a significant increase in efficiency over the older method of punching cards.

— effective for obtaining graphical displays of the results from the host computers. There was a significant increase in efficiency in analyzing results over the older method of looking at computer printout. (Many users did not invest the time required to convert from computer printout to graphical displays. However, the utility of graphical displays is becoming evident to most users and many are beginning to learn how to use the graphical packages.)

— somewhat effective for *dynamic* graphical displays. Users could store sequences of pictures on the local disks and then, while disconnected from the larger computers, display those pictures in a "movie" mode (at a rate of one picture per second) in order to examine the dynamic evolution of a system. The workbench was not capable of dynamically changing the view (e.g., via zoom, rotation, or translation) of the stored data.

— not very effective for viewing text output from the larger computers, because the workbench only permitted an 80-column width to be viewed on the CRT whereas user programs were normally designed to produce 132-column width output.

— not very effective for obtaining large text output files from the central computers, because of slow data transfer rates.

— not as effective for communicating with the VAX 11/780 computer as the VT100 terminals because some of the software on the VAX was written for the specific key layout of the VT100.

— effective for documentation of computer programs and for reporting results, because the editor was adequate for use as a word processor. (Users spent a large fraction of their time engaged in these two activities.)

— very effective for studying algorithms or small problems that could be executed directly on the microcomputer. However, most users did not utilize this feature extensively because the higher level languages on the workbench were not identical to the higher level languages on the larger computers (e.g., the FORTRANs were somewhat different) and because there was no package of scientific utility programs (such as the IMSL library) available on the workbench.

— availability was exceptional compared with the availability of time-shared systems. There was less than one failure per year on the average, and the time for repair was typically less than one day (because most repairs were done by board swapping). In addition, most users had access to more than one workbench.

— very effective for creating and modifying source code, provided the system was not saturated. There was a significant increase in efficiency over the older method of punching cards.

— very effective for viewing graphical output from computer models, when the terminals were equiped with a graphics modification. There was a significant increase in the efficiency of analyzing results over the older method of looking at computer printout.

— very effective for dynamically debugging modest size programs.

— effective for viewing text output from the computer models because VT100s can display a 132-column width, which corresponds to the standard line printer output format used in most programs.

— effective for documentation of computer programs and for reporting results, because the editor was adequate for use as a word processor.

— not effective for *dynamic* graphical displays because of slow data transfer rates.

§ 4.2 Factors that Impaired User Efficiency

The study illuminated many factors that prevented users from achieving optimum efficiency with these systems. The factors which inhibited *both* workbench and time-sharing system users from being more effective during this study were:

1. The initial effort required to convert from tabular output to graphical output was considered by many to be too time consuming.

2. The higher level languages (FORTRAN was the dominant language) were not highly portable. The users typically utilized the extensions provided for each computer so that fairly extensive modifications of the code were required to convert a program for use on a different computer. For example, users on the VAX 11/780 typically utilized the extensions provided by FORTRAN 77, so they were reluctant to transfer the programs to the CDC 7600 for long

production runs after the programs had been created and debugged on the VAX.

Factors which reduced workbench user efficiency for some users during this study were as follows:

1. The effort required to learn to use the system effectively was considered too great. The dialog interface to the workbench was too different from the dialog the user had learned in order to interface to the larger computers. (The amount of time a user is willing to spend to learn how to effectively utilize a local microcomputer varies greatly, but generally, users do not want to spend an appreciable time learning a completely different system.)

2. Documentation and training in the effective utilization of the microcomputers was not adequate for many users.

3. Local editing of very large files was not efficient, because the time required to transfer the files between the local microcomputer and the remote computer was excessive and these transfers could not be done as background tasks.

4. The ability to dynamically display data was limited to frame rates of one picture per second due to low processor speed and data transfer rates.

5. The complexity of the graphical displays was limited by the low resolution (320 dots horizontally by 240 dots vertically) graphics of the workbenches.

6. Users could not initiate one task on the workbench and then move on to a second task while the first continued.

Factors which reduced the efficient use of the time-sharing system were as follows:

1. The user response was degraded when the system became heavily loaded. For example, during text editing the response to cursor positioning became jerky and the user had to wait for the command buffer to empty in order to determine the actual cursor position. Also, under these conditions, search commands, which would normally appear to be executed instantaneously, could take 30 seconds to complete.

2. The user could not display graphics dynamically because the transfer rates to the terminal were inadequate.

3. The communications and computer systems for the time-sharing environment were much more complex and more frequently changed (upgraded) than the

single-user workbench system, and, therefore, the time-sharing environment had more downtime. In addition, there were normally no alternative systems available to time-sharing users when the central systems were down, so the availability of the time-shared systems was substantially less than the availability of the workbenches.

4. Data storage at remote computers was more difficult to manage than the local, single-user data storage of the workbench. The availability of the data stored at remote computers was substantially less than the availability of the data stored at the workbench for reasons described in item (3) above.

§ 5.0 Discussion

Before one can make recommendations based on the results above, one must consider how the software and the hardware are likely to change in the near future, which of the inhibiting factors listed above are likely to be eliminated by those changes, and which factors we will likely have to contend with for more than a few years.

Most of these factors should be eliminated by improvements forecast for the near future. Graphical display packages are becoming easier to use, and users are becoming more aware of the benefits of graphical displays. Transfer rates between remote systems and devices in the user's office are expected to increase to 10^6 to 10^7 bps utilizing nets such as Ethernet. Special hardware has been developed to permit good-resolution graphics with dynamic view manipulation. Systems for workbenches have been developed that permit users to work on several tasks simultaneously, and documentation for these systems is improving.

Those inhibiting factors that are likely to remain for more than a few years are as follows:

1. The initial effort required to learn how to use the system will probably be substantial. (The need to standardize and simplify the user enviroment has been recognized and efforts to accomplish this have begun — e.g., the DOD has initiated a project to establish a standard ADA programmer environment — but it will probably be several years before a standard is in common use.)

2. The higher-level languages will probably continue to have machine-dependent extensions that users will include in their codes so that codes will not be highly portable.

The inhibiting factors that are likely to remain for more than a few years for systems with only dumb terminals are listed below. Some of these are more related to human nature than to hardware or software limitations.

1. Sustained transfer rates from shared computers to local terminals will probably be too low to support dynamic graphics on dumb terminals. Although the transfer rate between time-shared computers and local equipment is expected to increase, it is doubtful that nets such as Ethernet could support many users attempting to obtain dynamical pictures simultaneously through the net. Workbenches with special graphics hardware are expected to offer much better dynamical graphics.

2. Shared systems will probably continue to become saturated. To date, all time-sharing systems that the authors have used have started with light computing loads and good response times and have evolved into heavily loaded systems with degraded response times.

3. Shared systems will continue to have lower availability than single user systems because they are more complex and more frequently modified.

4. Shared-storage systems will probably continue to have more management and availability problems than single-user systems.

§5.1 Discussion of Text Editing Software Because a large fraction of the user's time is spent creating and documenting programs and reporting results, factors which may inhibit the efficient use of software for text creation and modification are particularly important. Although the users of each of the hardware systems described in this report were supplied with screen-oriented, scrolling text editors which represented a significant improvement over any of the previously available technologies (e.g., keypunch or line-oriented text editors), each of the editors did have notable defficiencies in what might be termed the "human engineering" aspects of their designs.

Of the two editors, the VAX EDT editor, operating in the *keypad* mode, seemed to be preferred by most of the users who had experience with both. In the keypad mode, this editor offers a variety of rapid bi-directional scrolling and paging options, together with a selection of text-editing functions. Many of the complaints reported by users of this editor center on the lack of pre-defined commands to perform certain common functions. For example, there is no FIND AND REPLACE command *per se*; instead, the user must execute a sequence of three separate "atomic" functions to enter the replacement string, enter the target string, and find and replace the target string. (This problem is only partially remedied by the editor's *key re-definition* capability.)

The editor provided to users of the workbench was the UCSD Pascal System screen-oriented scrolling editor. Features available on this editor included bi-directional paging, and a selection of higher-level text editing functions. A principal drawback to the design of this editor seemed, from the user's viewpoint, to be the lack of an "immediate-insert" mode, in which the cursor movement controls and the text-delete function are isolated on the function keypad while all "printing" ASCII characters are inserted at the current cursor position by a single keystroke. Instead, user's were forced to select from a function menu in which the operations of text insertion and text deletion were completely separated, thereby adding the extra burden of frequent mode switching to the operational overhead of the editor.

In light of the preceding remarks, it seems clear that certain key recommendations can be made with a view toward achieving an overall improvement in user efficiency:

1. The same editor should be available on both the host and the local workstations.
2. The editor should be a full-screen, bi-directional, scrolling/paging text editor.
3. The editor should be initialized in the immediate insert mode, with separate keypad controls for cursor positioning and text deletion.
4. Some form of macro definition capability should be provided. The macro language should include branching and looping capabilities.
5. A text bracketing capability should be provided for the purpose of excerpting or deleting blocks of text.
6. A variety of high-level commands must be directly available. The user should not have to construct commonly used commands from more "atomic" commands.
7. The editor should have some facility for dealing with 132-character lines without wraparound.
8. Internal buffer management should be transparent to the user.
9. The editor should have safety features which provide some recourse in case of inadvertant text deletions.
10. File merge capabilities should be available.

§ 6.0 Recomendations

Providing software and hardware tools similar to those used in this study in order to improve the efficiency of scientists and programmers is highly recommended. Specific recommedations, based on the results of this study and technological changes forecast for the near future, are listed below.

<u>RECOMMENDED SOFTWARE TOOLS</u>

1. EDITOR

   — a general purpose screen oriented editor. (This tool was the most frequently used tool during the study. The significant features to look for are described in section 5.1 .)

2. DOCUMENTATION AND REPORTING UTILITIES

   — utilities to be used in conjunction with the editor to create documentation and reports. (For example, this report was formatted and typeset using the $\TeX$ technical typesetting system.)

3. JOB PREPARATION, SUBMITTAL, AND TRACKING UTILITIES

   — utilities for managing libraries and the configurations of the programs and data

   — a utility to check for program syntax errors that can be quickly detected

   — utilities for combining programs, data, and job control into a computational envelope

   — utilities for submitting the job envelope to a remote computer

   — utilities for tracking the progress of the job on the remote computer

4. POST PROCESSING UTILITIES

   — utilities for displaying results graphically

   — utilities for scanning text output

5. OPERATING SYSTEM

   — a command language that is preferably the same as the command language that the user employs on his other systems

— a multitasking facility so the user may have several tasks — e.g. a file transfer to a host computer and an editing task — being processed simultaneously

6. COMMUNICATIONS SOFTWARE

— dumb terminal emulation

— network virtual terminal support

— file transfer support

7. SELF DIAGNOSTIC UTILITIES

— utilities to check the hardware for defects and diagnose the nature and location of the defect

RECOMMENDED HARDWARE FEATURES

1. PROCESSING CAPABILITY

— processing power approximating that of a traditional midrange minicomputer — e.g. the processing power of an LSI 11/23, a Motorola 68000, or a Zilog Z8000 CPU

2. MEMORY CAPABILITY

— a primary memory of 1/4 megabyte or more

— a 5 megabyte or larger "Winchester" disk with backup capability

— an 8 inch floppy disk with standard format for physical transport of machine readable code and text files

3. DISPLAY CAPABILITY

— screen size of 12 inches or more

— text display 80 characters wide by 24 lines high; display switchable to 132-character width

— graphical resolution of 500 points or more horizontally with a square mesh

— ability to show dynamical displays (pictures with 1000 or more vectors displayed at a movie frame rate of one picture per second or faster)

4. COMMUNICATIONS CAPABILITY

   — interfaces to communicate to remote computers at transfer rates of 1 Mbps or more

5. INTERFACE WITH PROJECTION OR DISPLAY EQUIPMENT

   — video output for driving a standard TV monitor in TV seminar or conference rooms (or any other method for presenting the computer displays to groups of people)

6. HARDWARE TO OBTAIN HARDCOPY OF TEXT AND GRAPHS

   — a hardcopy unit with graphics capability for quicklook analysis located in the same room

   — a "publication quality" hardcopy unit located in the same building

7. PORTABILITY

   — the capability to easily move the unit from the office to the laboratory or to the home (At least the crucial components for doing off line work should be portable.)

§ 7.0 Conclusions

The use of a time-sharing computer system or the use of a local microcomputer as a workbench permitted users to work much more efficiently than they could by using punched cards for input and a line printer for computer output.

The study illuminated a number of factors which inhibited many users from utilizing all of the features of these systems effectively — particularly the features of the workbench. However, it seems reasonable to assume that, in light of projected improvements for these systems, most of these factors will be eliminated during the next few years, with a resulting increase in the effective utilization of these devices.

A primary problem for all users is the time consuming initial effort required in learning how to effectively utilize the systems. Although efforts are under way to standardize the user environment so that the user will not be required to learn a new command language, a new editor, etc., for each new system, a standard is not expected to be in common use in the next few years. To compensate for this situation, users will need adequate training and support.

Those problems expected to remain for several years in time-sharing systems with only dumb terminals are:

1. Transfer rates from shared systems to user sites are not expected to be high enough to permit effective dynamical graphics. Special graphical hardware at the user site will probably be necessary for effective dynamical graphics.

2. Shared systems are expected to continue to evolve into saturated systems with periods of poor response time.

3. Shared systems are expected to continue to have more data management and availability problems than single user data systems.

In order to compensate for these factors, the user should be provided with local processing for those activities for which response time is critical, such as editing and dynamical displays, and the user should be provided with local data storage for data that must be available at all times.

Specific software and hardware tools for improving scientist and programmer efficiency during the next few years were recommended.